UCRL-TR-202494

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Techniques for Judging Intent Behind Network Based Cyber Attacks

J. M. Allen

February 24, 2004

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

LDRD Final Report

Techniques for Judging Intent Behind Network Based Cyber Attacks.

Tracking Number: 03-ERD-012

Review and Release Tracking #: 304428

Author: John M. Allen, PI
Collaborators
Peter Balland
Ted Ferretta

Purpose

This project developed a prototype system that can rapidly differentiate between un-directed cyber attacks, and those that have a more specific and concerning intent behind them. The system responds to important cyber attacks in a tactically significant way as the attack is proceeding. It is also creates a prioritized list for the human analysts allowing them to focus on the threats mostly likely to be of interest. In the recent years the volume of attacks over the internet has increased exponentially, as they have become more and more automated. The result of this is that real threats are harder and harder to distinguish from the general threat. It is possible with our current systems to identify network packets that originated from thousands of IP addresses as probing a site like LLNL in a single day. Human analysis of these threats does not result in information that can be used for tactical response because most of the attacks are short and over before the human starts the analysis. Only a very small percentage of attacks can even be evaluated manually due to the volume. This project developed methods, and prototyped tools, that can identify attacks, slow the attack down and aid in the process of prioritizing detections. The project demonstrated that such methods exist, and that practical implementations exist for modern computers and networks. We call the tools created D.I.A.G. or Determining Internet Attackers Goals.

Approach

The approach we took was to create a system to simulate all or most of the hosts of a protected site and wait for an attack. The system engages the probing hosts by responding to their probes, and does so in a way that the attacker does not anticipate. The purpose of responding in an un-anticipated fashion, is to induce the attacker into betraying additional information about themselves and their intentions. We use this additional information to slow down the attack already in progress, as well as to help us classify the intent of the attacker. The goal is an automatic system to recognize skilled attackers and to tell the difference between those attackers and the high level of attack noise from worms and scripts not directed specifically at the protected site. One of the key ideas, is utilize our ability to generate patterns in the bogus data that we respond with that will be questioned by a skilled attacker, and ignored as spurious by a worm or un-attended script. One set carefully constructed results that we provide are crafted to appear as if there may have been the result of a malfunction. Transient malfunctions are not uncommon crossing the internet. The standard method of resolving such issues is to repeat the probe and compare results. Undirected attacks will ignore such spurious data and simply move on, a luxury not afforded to the attacker targeting a site that needs to understand every result. In essence, we are looking for indications that the attacker is implementing scientific principles of repeatability, to deduce the intent behind the attack. When we can detect these attempts to repeat the probe, we can identify those attacks with this level of analysis behind them. The time between the initial probe, and the attempts at establishing repeatability will also tell use something about how closely the probes are being monitored. To be practical to deploy at many sites a scaled up system would need to cost less than $100,000 per copy so we limited ourselves to designing around, and optimizing for an almost standard high end PC hardware. The system used cost less than $15,000 and supported targeted the prototype at a 1:4 scale. At this scale it was able to simulate 65535 IP addresses simultaneously with a mix of services on those hosts that is statistically indistinguishable from the real hosts at the protected site.

Accomplishments

1. We profiled several months of internet attack activity, obtaining network packet information from LLNL existing intrusion detection system. We extracted common patterns that were consistent over the dataset and used

them to tailor the database for a very high level of compression. In many cases this compression yields 100:1 or greater compression ratios over the time horizon of several months of data. This compression is important to creating a system that can simultaneously support fast access, maintain state for thousands of simulated hosts, and run on a system that is practical to deploy. This high level of compression allowed us to store much of the data on the prototype directly in memory where it could be accessed quickly enough to respond quickly enough to simulate how a real host would respond.

2. Fast insertions into the DB are also a necessary feature of keeping state for a large number of IP addresses. For purposes of the prototype we wanted to scale to a class B network, which can accommodate 65,535 unique IP addresses and include history for a large number of attackers. We sustained over 10,000 DB insertions per second, into our custom database on an $11,000 computer that was used for the prototype. This was accomplished by configuring the database to fit a statistical model of the average expected data patterns. This significantly exceeded the scalability goals for this aspect of the prototype and showed that a practical system, that would protect a large site like LLNL is possible utilizing off the shelf computing hardware.

3. We created a simulation of the LLNL yellow network. We wrote tools to take data collected at LLNL during system vulnerability scans, and we used that data as a statistical basis to create this prototype deception in such a way that in aggregate is blends with the real network because the instance of each host or application type in the deception is identical to their instances in the real network. This was also done to be sure that other un-anticipated scaling effects didn't exist in the database. It was successful, and a scan of the prototype simulation was statistically consistent with scans of the actual systems.

4. Wrote a tool to measure the OS/application timeouts in TCP connections. We accomplished this by simulating the actions of the operating system's network stack in our own code. We sent replies necessary to keep each connection open at the transport layer while sending nothing at the application layer. We measured the time between the initial connection and the last contact from the remote host. Using this method we were able to identify a number of different network scanning tools based upon their behavior.

5. We launched attacks against the prototype system. We compared the resulting output of the attack tools to that that we expected them to receive. In these laboratory conditions we were able to sustain network traffic volume equal to that of a large Internet site like LLNL, assuming that 80% of the traffic was attack traffic which is well above expected limits as this system is not designed to deal with denial of service attacks. A collection of common hacker tools all returned our crafted results in the expected fashion. We

compared the scan results against the deceptions, with the scan results of actual systems.

6. We put the prototype out on the Internet to see what happens against real world conditions. The prototype was given a couple of IP addresses, and the most basic of our deceptions. The system was allowed to run for 20 days. During that time, 11745 unique SRC IP’s sent some sort of un-solicited network probes to 1 or more of the IP addresses used by the prototype system. Of those detects, 9 turned out to be authorized systems, that were expected to probe all IP’s at the protected site, leaving 11736 probes for which we did not know the intent. One of the hypothesis of the project is that Internet worms would just ignore our strange but legal responses to their probes, but that an intelligent attacker would retry the probe, and betray their existence at the other end, in an attempt to understand the confusing output. We sorted the 11736 detects by the quantity of traffic that they involved. The 5^{th} IP on the list, was a clear success. Human analysis of this detect by both the research team, as well as other intrusion analysts to which the detect was passed reached the unanimous conclusion that it was clearly a skilled, knowledgeable attacker, that was monitoring the results of their attack in real time, and applying scientific principles of repeatability to attempt to determine if the validity of the responses they got from the prototype system. The same attacking system that interacted with the prototype system, also interacted with a number of real systems, which were used as the control group in this phase of this experiment. When it interacted with one of the control hosts, the probes appeared to be just a standard worm type of scan. It would hit 1 system every 60-120 minutes, make a single connection, and a single probe inside that connection. It appears to pick IP addresses at random. Traditional signature based IDS systems did not alert on any of the contents of the packets to the control hosts. A honey pot system, would have registered the scan, but been unable to differentiate this scan from a worm. Protocol analysis IDS systems would not have triggered on the activity to the control host either, as the probe was a legit from that perspective as well.

   A Comparison of the skilled attacker between a regular control host and the LDRD prototype with the most basic of it’s deceptions.

Attack on a real host

Connection sequence done once
One source port used
8 packets sent over two seconds
No known signature based alerts triggered
Looks exactly like an Internet worm to both automated and human analyst.

Attack against prototype protected IP

Connection repeated 4 times
8 different source ports used

37 packets sent over 33 minutes and 52 seconds
1 Signature based alert triggered 4 times

Clearly an attack by a skilled attacker who is looking at the results generated, and who reacts in a knowledgeable and logical fashion including using scientific principles to interpret the responses we provided to their probes.

Conclusion

The LDRD's opening premise, is valid. It is possible to supply both the worms of the internet, and the human hackers un-expected, but completely legit responses to their probes, and then tell the difference between the two by the way they react or don't react to that specially crafted data. The second premise, that we could develop something that would also slow an attacker down enough to constitute a tactical defense is also valid. Not only did it work in controlled conditions, in the test on the live internet it took an attack that would have lasted only 2 seconds and stretched it out to in excess of 33 minutes. The final premise, that this might even be practical to implement also turned out to be correct. On a borrowed $11,000 computer we were able to simulate a network 1/4 the size of LLNL's network with a basic deception which is well under what sites like LLNL pay for other types of intrusion detection devices.